UNITED STATES PATENT AND TRADEMARK OFFICE

UNITED STATES DEPARTMENT OF COMMERCE
United States Patent and Trademark Office
Address: COMMISSIONER FOR PATENTS
P.O. Box 1450
Alexandria, Virginia 22313-1450
www.uspto.gov

APPLICATION NO.	FILING DATE	FIRST NAMED INVENTOR	ATTORNEY DOCKET NO.	CONFIRMATION NO.
10/820,661	04/07/2004	Paul A. Martin	SUN04-0234	8024

57960 7590 03/11/2008

SUN MICROSYSTEMS INC.
C/O PARK, VAUGHAN & FLEMING LLP
2820 FIFTH STREET
DAVIS, CA 95618-7759

EXAMINER	
KIM, PAUL	

ART UNIT	PAPER NUMBER
2161	

MAIL DATE	DELIVERY MODE
03/11/2008	PAPER

Please find below and/or attached an Office communication concerning this application or proceeding.

The time period for reply, if any, is set in the attached communication.

Office Action Summary	Application No.	Applicant(s)	
	10/820,661	MARTIN, PAUL A.	
	Examiner	Art Unit	
	Paul Kim	2161	

-- The MAILING DATE of this communication appears on the cover sheet with the correspondence address --

Period for Reply

A SHORTENED STATUTORY PERIOD FOR REPLY IS SET TO EXPIRE *3* MONTH(S) OR THIRTY (30) DAYS, WHICHEVER IS LONGER, FROM THE MAILING DATE OF THIS COMMUNICATION.

- Extensions of time may be available under the provisions of 37 CFR 1.136(a). In no event, however, may a reply be timely filed after SIX (6) MONTHS from the mailing date of this communication.
- If NO period for reply is specified above, the maximum statutory period will apply and will expire SIX (6) MONTHS from the mailing date of this communication.
- Failure to reply within the set or extended period for reply will, by statute, cause the application to become ABANDONED (35 U.S.C. § 133). Any reply received by the Office later than three months after the mailing date of this communication, even if timely filed, may reduce any earned patent term adjustment. See 37 CFR 1.704(b).

Status

1)☒ Responsive to communication(s) filed on *28 November 2007*.

2a)☐ This action is **FINAL**. 2b)☒ This action is non-final.

3)☐ Since this application is in condition for allowance except for formal matters, prosecution as to the merits is closed in accordance with the practice under *Ex parte Quayle*, 1935 C.D. 11, 453 O.G. 213.

Disposition of Claims

4)☒ Claim(s) *1,3-15,17-29 and 31-42* is/are pending in the application.

4a) Of the above claim(s) _____ is/are withdrawn from consideration.

5)☐ Claim(s) _____ is/are allowed.

6)☒ Claim(s) *1,3-15,17-29 and 31-42* is/are rejected.

7)☐ Claim(s) _____ is/are objected to.

8)☐ Claim(s) _____ are subject to restriction and/or election requirement.

Application Papers

9)☐ The specification is objected to by the Examiner.

10)☐ The drawing(s) filed on _____ is/are: a)☐ accepted or b)☐ objected to by the Examiner.

Applicant may not request that any objection to the drawing(s) be held in abeyance. See 37 CFR 1.85(a).

Replacement drawing sheet(s) including the correction is required if the drawing(s) is objected to. See 37 CFR 1.121(d).

11)☐ The oath or declaration is objected to by the Examiner. Note the attached Office Action or form PTO-152.

Priority under 35 U.S.C. § 119

12)☐ Acknowledgment is made of a claim for foreign priority under 35 U.S.C. § 119(a)-(d) or (f).

a)☐ All b)☐ Some * c)☐ None of:

1.☐ Certified copies of the priority documents have been received.

2.☐ Certified copies of the priority documents have been received in Application No. _____.

3.☐ Copies of the certified copies of the priority documents have been received in this National Stage application from the International Bureau (PCT Rule 17.2(a)).

* See the attached detailed Office action for a list of the certified copies not received.

Attachment(s)

1) ☒ Notice of References Cited (PTO-892)
2) ☐ Notice of Draftsperson's Patent Drawing Review (PTO-948)
3) ☐ Information Disclosure Statement(s) (PTO/SB/08) Paper No(s)/Mail Date _____.
4) ☐ Interview Summary (PTO-413) Paper No(s)/Mail Date. _____ .
5) ☐ Notice of Informal Patent Application
6) ☐ Other: _____.

DETAILED ACTION

1. This Office action is responsive to the following communication: Request for Continued Examination filed on 31 August 2007.

2. Claims 1-42 are pending and present for examination. Claims 1, 15, and 29 are in independent form.

Continued Examination Under 37 CFR 1.114

3. A request for continued examination under 37 CFR 1.114, including the fee set forth in 37 CFR 1.17(e), was filed in this application after final rejection. Since this application is eligible for continued examination under 37 CFR 1.114, and the fee set forth in 37 CFR 1.17(e) has been timely paid, the finality of the previous Office action has been withdrawn pursuant to 37 CFR 1.114. Applicant's submission filed on 31 August 2007 has been entered.

Response to Amendment

4. Claims 1, 3-5, 9-10, 12, 15, 17-19, 23-24, 26, 29, 31-33, 37-38, and 40 have been amended.

5. Claims 2, 16, and 30 have been cancelled.

6. No claims have been further added.

Claim Rejections - 35 USC § 103

7. The following is a quotation of 35 U.S.C. 103(a) which forms the basis for all obviousness rejections set forth in this Office action:

> (a) A patent may not be obtained though the invention is not identically disclosed or described as set forth in section 102 of this title, if the differences between the subject matter sought to be patented and the prior art are such that the subject matter as a whole would have been obvious at the time the invention was made to a person having ordinary skill in the art to which said subject matter pertains. Patentability shall not be negatived by the manner in which the invention was made.

8. **Claims 1, 4, 6-9, 11, 13-14, 15, 18, 20-23, 25, 27-28, 29, 32, 34-37, 39, and 41-42** are rejected under 35 U.S.C. 103(a) as being unpatentable over McGregor (NPL, "Practical C++, published by Que on 11 August 1999) in view of Guthrie, II (U.S. Patent No. 7,225,210, hereinafter referred to as Guthrie), filed on 20 November 2003, published on 26 May 2005, and issued on 29 May 2007.

9. **As per independent claims 1, 15 and 29,** McGregor, in combination with Guthrie, discloses:

A method for performing a lock-free update to one or more fields in an existing node in a linked list, comprising:

receiving a reference to the existing node in the linked list, wherein the existing node contains the one or more fields to be updated {See McGregor, page 8, wherein this reads over "start at either the head or the tail and traverse the list until the item is found"};

obtaining a new node to be added to the linked list, wherein other processes do not possess references to the new node and therefore cannot initially access the new node {See McGregor, page 9, wherein this reads over "allocated a new node"};

copying a snapshot of the existing node to the new node {See Guthrie, col. 8, lines 20-67, wherein this reads over "the snapshot system created a new node 6 and incremented the snapshot identifier of node 0 to 1"}, which includes copying a next pointer of the existing node to the new node, so that the new node points to a node immediately following the existing node {See McGregor, page 7, wherein this reads over "the address of the new node is assigned to a node pointer"; and page 11, wherein this reads over "all you have to do is swap the pointers of the nodes on either side of the node you are deleting"};

updating one or more fields in the new node that correspond to the one or more fields in the existing node {See McGregor, page 9, wherein this reads over "set the integer value for the node"};

performing a single atomic operation that modifies the next pointer of the existing node to point to the new node and also marks the next pointer to indicate that the existing node is deleted, whereby the new node becomes part of the linked list and the existing node is deleted in a single atomic operation {See McGregor, page 11, wherein this reads over "all you have to do is swap the pointers of the nodes on either side of the node you are deleting"}; and

splicing the existing node out of the linked list by atomically modifying the next pointer of a node immediately preceding the existing node in the linked list to point to the new node, instead of pointing to the existing node {See McGregor, page 11, wherein this reads over "all you have to do is swap the pointers of the nodes on either side of the node you are deleting"}.

While McGregor may fail to expressly disclose the copying of a snapshot of an existing node, Guthrie discloses a snapshot data system wherein snapshots of nodes comprising a plurality of fields are taken and copied. Accordingly, the combination of invention disclosed by McGregor and Guthrie would

disclose an invention which would comprise a method for obtaining a snapshot of an existing node and applying said copy to a newly created node such that updates may be made to the newly created node without necessitating a lock of the existing node. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the above invention suggested by McGregor by combining it with the invention disclosed by Guthrie.

One of ordinary skill in the art would have been motivated to do this modification such that a newly created node, which consists of copied fields which have been updated, may be inserted into a linked list.

10. **As per dependent claims 4, 18, and 32,** McGregor, in combination with Guthrie, discloses:

The method of claim 1, wherein copying a snapshot of the existing node to the new node involves:

copying the contents of the existing node to the new node {See Guthrie, col. 8, lines 43-54, wherein this reads over "[t]he snapshot system copied the data of root node 0 to the root node 6 of the snapshot"};

examining the next pointer of the existing node to determine if the existing node has been deleted {See McGregor, page 8, wherein this reads over "CIntList::Find( ) function"}; and

if so taking a remedial action;

otherwise, not taking the remedial action.

11. **As per dependent claims 6, 20, and 34,** McGregor, in combination with Guthrie, discloses:

The method of claim 1, further comprising deleting a target node from the linked list by:

receiving a reference to the target node to be deleted from the linked list {See McGregor, page 8, wherein this reads over "CIntList::Find( ) function"};

atomically marking a next pointer in the target node to indicate that the target node is deleted {See McGregor, page 11, wherein this reads over "all you have to do is swap the pointers of the nodes on either side of the node you are deleting"}; and

atomically modifying the next pointer of a node immediately preceding the target node in the linked list to point to a node immediately following the target node in the linked list, instead of pointing to the target node, thereby splicing the target node out of the linked list {See McGregor, page 11, wherein this reads over "all you have to do is swap the pointers of the nodes on either side of the node you are deleting"}.

12. **As per dependent claims 7, 21, and 35,** McGregor, in combination with Guthrie, discloses:

The method of claim 6, wherein after the target node is spliced out of the linked list, the method further comprises modifying the next pointer of the target node so that the next pointer remains marked but points to a node immediately preceding the target node instead of the node immediately following node the target node in the linked list {See McGregor, page 11, wherein this reads over "all you have to do is swap the pointers of the nodes on either side of the node you are deleting"}.

13. **As per dependent claims 8, 22, and 36,** McGregor, in combination with Guthrie, discloses:

The method of claim 1, further comprising inserting an additional node into the linked list by:

identifying a node immediately preceding the additional node in the linked list {See McGregor, page 8, wherein this reads over "CIntList::Find( ) function";

identifying a node immediately following the additional node in the linked list {See McGregor, page 8, wherein this reads over "CIntList::Find( ) function"; and

splicing the additional node into the linked list by, setting the next pointer for the additional node to point to the immediately following node {See McGregor, page 11, wherein this reads over "all you have to do is swap the pointers of the nodes on either side of the node you are deleting"}, and

atomically updating the next pointer of the immediately preceding node to point to the additional node {See McGregor, page 11, wherein this reads over "all you have to do is swap the pointers of the nodes on either side of the node you are deleting"}.

14. **As per dependent claims 9, 23, and 37,** McGregor, in combination with Guthrie, discloses:

The method of claim 1, further comprising

reading a snapshot of multiple fields from a target node in the linked list by:

reading the multiple fields from the target node {See Guthrie, col. 8, lines 20-38, wherein this reads over "[t]emplate 100 illustrates the fields of the node"};

examining the next pointer of the target node to determine if the target node has been deleted {See McGregor, page 8, wherein this reads over "CIntList::Find( ) function"}; and

if so, taking a remedial action;

otherwise, not taking the remedial action.

15. **As per dependent claims 11, 25, and 39,** McGregor, in combination with Guthrie, discloses:

The method of claim 1, wherein atomically modifying the next pointer of the existing node to indicate that the existing node is deleted involves setting a "deleted bit" in the next pointer {See McGregor, page 11, wherein this reads over "reset the node's address to zero"}.

16. **As per dependent claims 13, 27, and 41,** McGregor, in combination with Guthrie, discloses:

The method of claim 1, wherein a given node in the linked list includes:

a key that contains an identifier for the given node {See McGregor, page 4, wherein this reads over "Find ( ) Returns a pointer to the first node containing the specified integer value"};

one or more fields containing data values or pointers to data values associated with the given node {See Guthrie, col. 8, lines 20-38, wherein this reads over "[t]emplate 100 illustrates the fields of the node"}; and

a next pointer that contains the address of a node that immediately follows the given node in the linked list, and that also contains a deleted indicator, which indicates whether the given node has been deleted {See McGregor, page 7, wherein this reads over "the address of the new node is assigned to a node pointer"; and page 11, wherein this reads over "all you have to do is swap the pointers of the nodes on either side of the node you are deleting"}.

17. **As per dependent claims 14, 28, and 42,** McGregor, in combination with Guthrie, discloses:

The method of claim 1, further comprising periodically performing a garbage-collection operation to reclaim deleted nodes that have become unreachable {See McGregor, page 11, wherein this reads over "[d]elete the node (which has now been unlinked from the list), reset the node's address to zero"}.

18. **Claims 3, 17, and 31** are rejected under 35 U.S.C. 103(a) as being unpatentable over McGregor, in view of Guthrie, and further in view of Official Notice.

19. **As per dependent claims 3, 17, and 31,** the Examiner takes Official Notice that it would have been obvious and widely-known to one of ordinary skill in the art to execute a splicing process should a prior process fail to perform the splicing operation.

20. **Claims 5, 10, 12, 19, 24, 26, 33, 38, and 40** are rejected under 35 U.S.C. 103(a) as being unpatentable over McGregor, in view of Guthrie, and further in view of Lippman et al (NPL, "C++ Primer," published by Addison Wesley Professional on 2 April 1998.

21. **As per dependent claims 5, 19, and 33,** McGregor, in combination with Guthrie and Lippman, discloses:

The method of claim 4, wherein taking the remedial action involves:

following the next pointer of the existing node in an attempt to find an updated version of the existing node {See McGregor, page 8, wherein this reads over "CIntList::Find( ) function"};

if an updated version of the existing node is found, copying a snapshot of the updated version of the existing node to the new node {See Guthrie, col. 8, line 55 – col. 9, line 9, wherein this reads over "[t]he snapshot system then created a new node for the node being modified" and "set the snapshot identifier field of node 8 to 2 and set the previous field of node 8 to 2"}; and

if an updated version of the existing node is not found, indicating that the remedial action fails {See Lippman, page 5, wherein this reads over "assert ( )"}.

While McGregor and Guthrie may fail to expressly disclose a method of providing an indication of a remedial action failure, Lippman discloses a method for announcing a condition that triggers the assertion. Accordingly, the combination of invention disclosed by McGregor, Guthrie, and Lippman would disclose an invention which would comprise a method for announcing a triggering condition. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the above invention suggested by McGregor and Guthrie by combining it with the invention disclosed by Lippman.

One of ordinary skill in the art would have been motivated to do this modification such that an indication may be provided that a certain remedial action failed.

22. **As per dependent claims 10, 24, and 38,** McGregor, in combination with Guthrie and Lippman, discloses:

The method of claim 9, wherein taking the remedial action involves:

following the next pointer of the target node in an attempt to find an updated version of the target node {See McGregor, page 8, wherein this reads over "CIntList::Find( ) function"}; and

if an updated version of the target node is found, repeating the process of reading a snapshot of the multiple fields from the updated version of the target node {See Guthrie, col. 8, line 55 – col. 9, line 9, wherein this reads over "[t]he snapshot system then created a new node for the node being modified" and "set the snapshot identifier field of node 8 to 2 and set the previous field of node 8 to 2"}; and

if an updated version of the existing node is not found, indicating that the remedial action fails {See Lippman, page 5, wherein this reads over "assert ( )"}.

While McGregor and Guthrie may fail to expressly disclose a method of providing an indication of a remedial action failure, Lippman discloses a method for announcing a condition that triggers the assertion. Accordingly, the combination of invention disclosed by McGregor, Guthrie, and Lippman would

disclose an invention which would comprise a method for announcing a triggering condition. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the above invention suggested by McGregor and Guthrie by combining it with the invention disclosed by Lippman.

One of ordinary skill in the art would have been motivated to do this modification such that an indication may be provided that a certain remedial action failed.

23. **As per dependent claims 12, 26, and 40,** McGregor, in combination with Guthrie and Lippman, discloses:

> The method of claim 1, wherein while atomically modifying the next pointer of the existing node,
>
> if the next pointer indicates that the existing node is already deleted, the atomic modification operation fails and the method further comprises taking a remedial action to deal with the fact that the existing node is already deleted {See Lippman, page 5, wherein this reads over "assert ( )"};
>
> otherwise, continuing performing the atomic modification operation.

While McGregor and Guthrie may fail to expressly disclose a method of providing an indication of a remedial action failure, Lippman discloses a method for announcing a condition that triggers the assertion. Accordingly, the combination of invention disclosed by McGregor, Guthrie, and Lippman would disclose an invention which would comprise a method for announcing a triggering condition. Therefore, it would have been obvious to one of ordinary skill in the art at the time the invention was made to modify the above invention suggested by McGregor and Guthrie by combining it with the invention disclosed by Lippman.

One of ordinary skill in the art would have been motivated to do this modification such that an indication may be provided that a certain remedial action failed.

Conclusion

24. Any inquiry concerning this communication or earlier communications from the examiner should be directed to PAUL KIM whose telephone number is (571)272-2737. The examiner can normally be reached on M-F, 9am - 5pm.

If attempts to reach the examiner by telephone are unsuccessful, the examiner's supervisor, Apu Mofiz can be reached on (571) 272-4080. The fax phone number for the organization where this application or proceeding is assigned is 571-273-8300.

Information regarding the status of an application may be obtained from the Patent Application Information Retrieval (PAIR) system. Status information for published applications may be obtained from either Private PAIR or Public PAIR. Status information for unpublished applications is available through Private PAIR only. For more information about the PAIR system, see http://pair-direct.uspto.gov. Should you have questions on access to the Private PAIR system, contact the Electronic Business Center (EBC) at 866-217-9197 (toll-free). If you would like assistance from a USPTO Customer Service Representative or access to the automated information system, call 800-786-9199 (IN USA OR CANADA) or 571-272-1000.

/Apu M Mofiz/
Supervisory Patent Examiner, Art Unit 2161

Paul Kim
Examiner, Art Unit 2161
TECH Center 2100